

## The JET Intershot Analysis: Current infrastructure and future plans

R. Layne<sup>a,b,c,\*</sup>, N. Cook<sup>a,b,c</sup>, D. Harting<sup>a,d</sup>, D.C. McDonald<sup>a,b</sup>, C. Tidy<sup>a,e</sup>, JET EFDA contributors<sup>1</sup>

<sup>a</sup> JET-EFDA, Culham Science Centre, OX14 3DB, Abingdon, UK

<sup>b</sup> EURATOM-UKAEA Fusion Association, Culham Science Centre, Abingdon, Oxon OX14 3DB, UK

<sup>c</sup> Tessella plc, 26 The Quadrant, Abingdon Science Park, Abingdon, Oxon OX14 3YS, UK

<sup>d</sup> Institute for Energy Research IEF-4, Forschungszentrum Jülich, Association Euratom-FZJ, Trilateral Euregio Cluster, Germany

<sup>e</sup> University of Bath, Computer Science, BA2 7AY Bath, UK

### ARTICLE INFO

#### Article history:

Available online 18 January 2010

#### Keywords:

Integrated  
Analysis  
Workflow

### ABSTRACT

The JET Intershot Analysis (Chain1) generates processed data following a pulse. Maintaining the pulse repetition rate is one of JET's key success factors, so performance of Chain1 is crucial. This paper will describe JET's experience of managing Chain1, including a description of the control system used to ensure the analysis chain runs as quickly as possible, and a discussion of JET's experience of integrating externally developed codes into a standard analysis framework.

The current Chain1 infrastructure was developed in 1999 and although reliable and efficient is starting to prove costly in terms of flexibility and extensibility to meet JET's current and future needs. For this reason JET is planning to re-implement the Chain1 system. The paper will outline the work done towards this aim, and present a model of the proposed new system. Finally, possible future steps towards an integrated data production chain for JET will be discussed, and the potential applicability to next generation fusion devices will be outlined.

© 2010 EURATOM. Published by Elsevier B.V. All rights reserved.

### 1. Introduction

The Intershot Analysis, also known as Chain1, is a crucial part of JET's operational data management cycle [1]. Chain1 is responsible for analysing the raw data collected during a JET pulse to produce processed data (see Fig. 1). The processed data is often required by experimentalists to plan the following pulse, so it is important that Chain1 completes its processing as quickly as possible. Data quality is fundamental to JET's mission, so the quality and integrity of the data produced must be ensured.

Chain1 currently consists of 85 analysis codes, many developed by external developers, with complex dependencies on each other and around 60 raw data sources. A task scheduler has been developed at JET to ensure that these codes run as quickly as possible to maintain the JET pulse repetition rate. Procedures have been developed to control integration of analysis codes and configuration data to ensure the quality of the output data.

The current Chain1 infrastructure has worked well but has a number of limitations for the current and future needs of the project. As a result, a phased series of enhancements to the control

system are underway, which will ultimately lead to a fully integrated data production workflow for the processed experimental data.

### 2. The current Chain1 infrastructure

The current Chain1 system was written in the late 1990s, based on the original design of the system from the mid-1980s [2].

There are two main components:

1. Analysis codes, which may be developed at other research organisations to be integrated into the processing chain.
2. The task scheduler, which runs the codes and monitors the performance of the chain.

#### 2.1. Analysis codes

Chain1 currently consists of 85 independent analysis codes (called "steps"), which use input data from around 60 raw data sources from plant and diagnostics. The collection time for the raw data sources is variable, so some are available earlier than others. Each of the steps depends on raw data, and in most cases on the input from one or more other steps. This leads to a complex set of interdependencies.

Fig. 2 shows a small part of the processing chain. Items in circles are JET's raw data collecting subsystem computers, all of which have two letter identifiers, for instance PF. Once all raw data sources

\* Corresponding author at: EURATOM-UKAEA Fusion Association, Culham Science Centre, Abingdon, Oxon OX14 3DB, UK. Tel.: +44 01235 465021; fax: +44 01235 464404.

E-mail address: [richard.layne@ccfe.ac.uk](mailto:richard.layne@ccfe.ac.uk) (R. Layne).

<sup>1</sup> See the Appendix of F. Romanelli et al., Fusion Energy Conference 2008 (Proc. 22nd Int. FEC Geneva, 2008), IAEA, 2008.

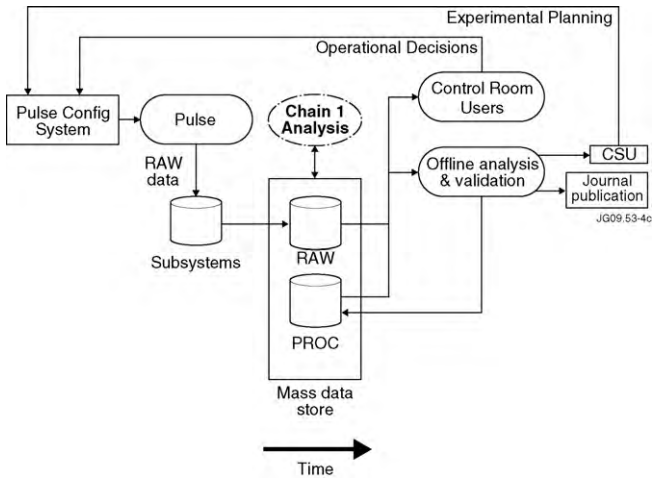


Fig. 1. Chain1 in the context of JET's data management cycle.

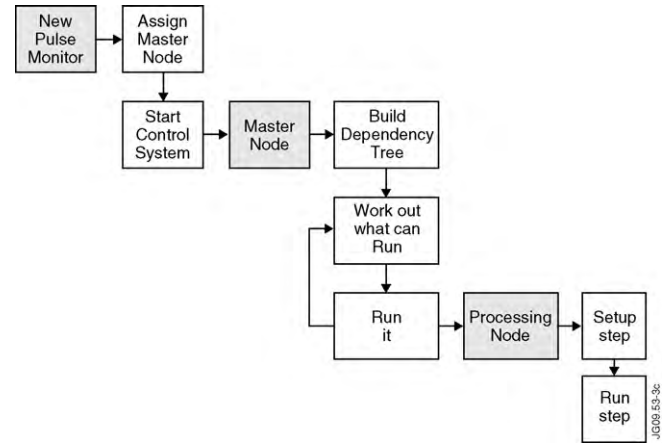


Fig. 3. Overview of the current Chain1 scheduler.

on a particular subsystem have been collected, the data is available to Chain1. Items in rectangles are Chain1 steps, all of which have one to four letter identifiers (for instance, NBI calculates the Neutral Beam Power). Arrows show dependencies between steps; dotted arrows indicate dependencies on raw data sources. The dashed line at the top right indicates that FAST (a step which determines physical parameters of the plasma) depends on either EFIT (a step which carries out plasma shape, equilibrium and flux surface reconstruction based on magnetics data) or XLOC (which carries out low resolution plasma boundary determination based on magnetics data).

A flexible framework for integration of new codes has been developed to aid external developers. All codes must run in batch with a defined, fixed interface, and must require no manual intervention. Software quality is ensured with standards defined for supported languages, coding standards, documentation, input file structure, error codes, and output data format.

Quality of the output data is ensured by a peer review process; example data must be produced and the new code must be presented at the JET Data Validation and Coordination Meeting for approval before it is integrated into the chain. Once integrated, source code and input files are subject to configuration management in a CVS (Concurrent Versions System) [3] repository.

2.2. Task scheduler

The Chain1 task scheduler has two main components, written in C and Perl [4]. An overview is shown in Fig. 3.

The New Pulse Monitor is a background process, which runs continuously, waiting for notification of a new pulse from JET's Object Monitoring System [5]. On notification, it obtains the pulse date and time, creates the new pulse in the JET processed data storage (PPF) system [6], and starts the task control system which will run Chain1 for the new pulse. The task control system runs on an analysis node known as the "Master Node".

The task control system manages Chain1 execution for an individual pulse. A dependencies file indicates which raw and processed data sources are required for each step. The system monitors the arrival of the input raw data sources, and the completion of each analysis code. Once the required data for a particular step is present, the step is submitted. The system aims to balance processing between twelve dedicated Linux machines in the JET Analysis Cluster (JAC), spawning each step to the least loaded machine using rsh. The JAC cluster consists of 142 nodes with a total of 336 Athlon processor cores, using the Linux operating system Fedora Core 10 [7].

2.3. Advantages and disadvantages of the current system

The current system is relatively simple, it has worked well since 2000, and Chain1 processing is consistently fast despite the growth

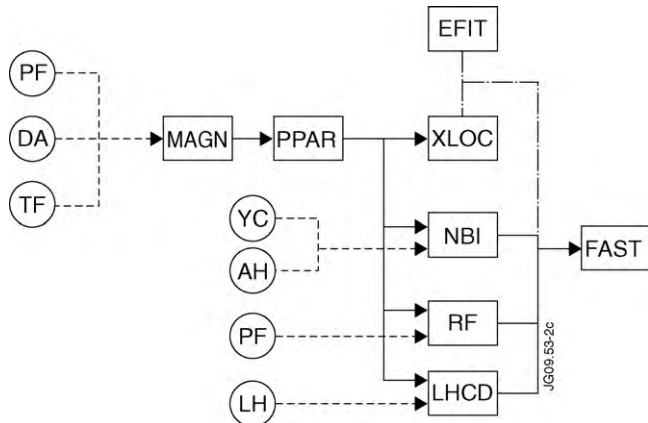


Fig. 2. Example of raw and processed data dependencies.

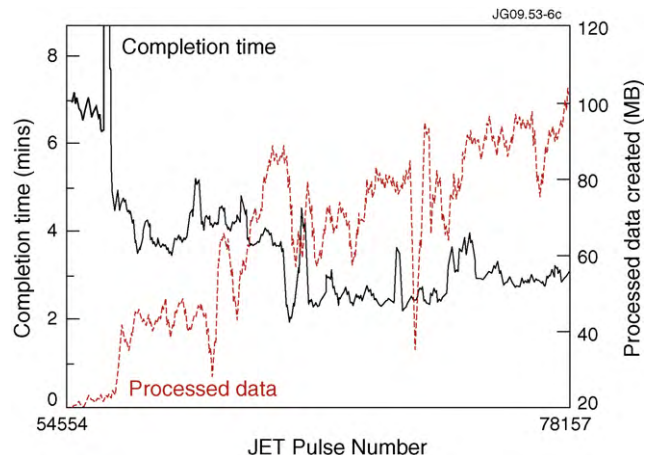


Fig. 4. Chain1 completion time and JET data creation.

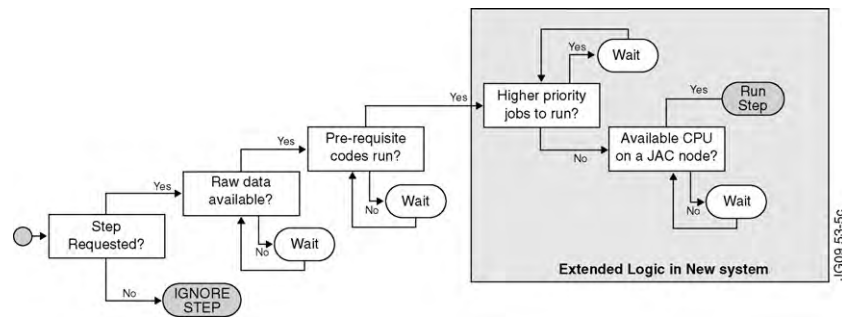


Fig. 5. Enhanced processing logic in the new system.

in data production, as shown in Fig. 4 which shows Chain1 completion time and the volume of data created, for JET plasma pulses between pulse 54,554 (June 2002) and 78,157 (April 2009).

Our approach to integration of new codes ensures both the quality of the output data, and the software quality of the analysis code using the expertise of physicists and software developers. Further, the team supporting Chain1 includes both computational physicists and software engineers: the expertise of both is crucial to the continued successful operation of the processing chain.

However, there are a number of limitations.

- The task scheduler is inflexible, and does not reflect the true complexity of the dependencies between the steps.
- There is no control over a step once it has been submitted to a JAC node.
- The view of machine load is based purely on the number of Chain1 steps currently running on each machine. This does not take account of the needs of each step throughout its run. Retrospective analysis of performance statistics of the processing nodes shows peaks and troughs in CPU usage, showing our load balancing may not be optimal.
- Some steps with high CPU requirements are delayed to the end of the chain using “false” dependencies. This is not a transparent or flexible approach.
- Traceability of processed data back to the analysis code version which created it is not complete, and it is time consuming to revert to earlier versions of the chain.
- Including new steps and updates in the chain requires a lot of work from the support team.

For these reasons, development of a new Chain1 infrastructure is currently underway, which will be outlined in the following sections. This development is being carried out in phases, to allow JET to benefit quickly from each new development.

### 3. Phase 1: new infrastructure for next JET campaign

The first step towards the new infrastructure is a new control system, which will be released for use in the next JET experimental campaign (Summer 2009), and a new Chain1 performance monitoring and modelling system which has been in use since 2008.

#### 3.1. New task scheduler

The new task scheduler, written, like the current system, in C, supports better encapsulation of Chain1 entities (i.e. raw data sources, steps and JAC nodes), to aid extensibility and maintainability. Each entity is now treated as a structure instance, with all related data within it. Linked lists are used to traverse relationships both within and between structures.

In the current release system, monitoring of raw data creation was done using a hard coded list of data sources, so had to be re-released every time a new raw data source was added. The list of data sources is now determined dynamically at run-time from a configuration file. The new system will also monitor a wider range of raw data sources – previously only the JPF (JET Pulse File) was monitored: the new system will also take account of the earlier arrival of the IPF (Immediate Pulse File) and the QPF (Quick Pulse File) and the later arrival of the DPF (Delayed Pulse File) and LPF (Late Pulse File).

Enhanced processing logic is supported in the new system, as shown in Fig. 5. New rules will allow steps to be run on specific nodes, based on their known maximum CPU requirements, so avoiding 100% CPU loading on a particular node. This approach also allows us to separate running of single node and parallel jobs. JET’s PPF system [6] allows users to create “private” test data as well as the official “public” PPF data. Public and private data sources are now supported for input PPF data which will significantly improve our ability to run the processing chain in experimental ways when testing existing or evaluating new analysis codes.

In the current system, “false” dependencies are used to force some steps to run later, for instance for steps which require high CPU usage but which are not on the Chain1 critical path. This is now implemented using a priority-based system.

The previous system used two configuration files to specify which steps could run and when. This has been replaced with a single file accessed once to build a dynamic dependency list.

#### 3.2. Chain1 monitor software

An earlier version of the Chain1 monitor software showed a snapshot of Chain1 status at a given time within its run [8]. This has been replaced with an enhanced version, written in PHP [9], which, along with the snapshot view, displays a visual representation of step performance, data relationships and analysis progress, in a dynamic Gantt Chart view (see Fig. 6). This can be viewed in pseudo real-time, for the current pulse, and an archive of performance for previous runs is available.

The new monitor can therefore be used as a display tool to inform experimentalists of progress during the current experiment, and to notify them when data items are available. It also has a secondary use as an analysis tool to improve system efficiency and overall Chain1 performance.

The Chain1 administrators can use the tool to analyse the structure of the chain and identify potential bottlenecks and areas to target for improvement. It also provides evidence that can be presented to analysis code owners to show them the impact their code has on overall Chain1 performance, and demonstrate whether a change to their code has improved or worsened its performance.

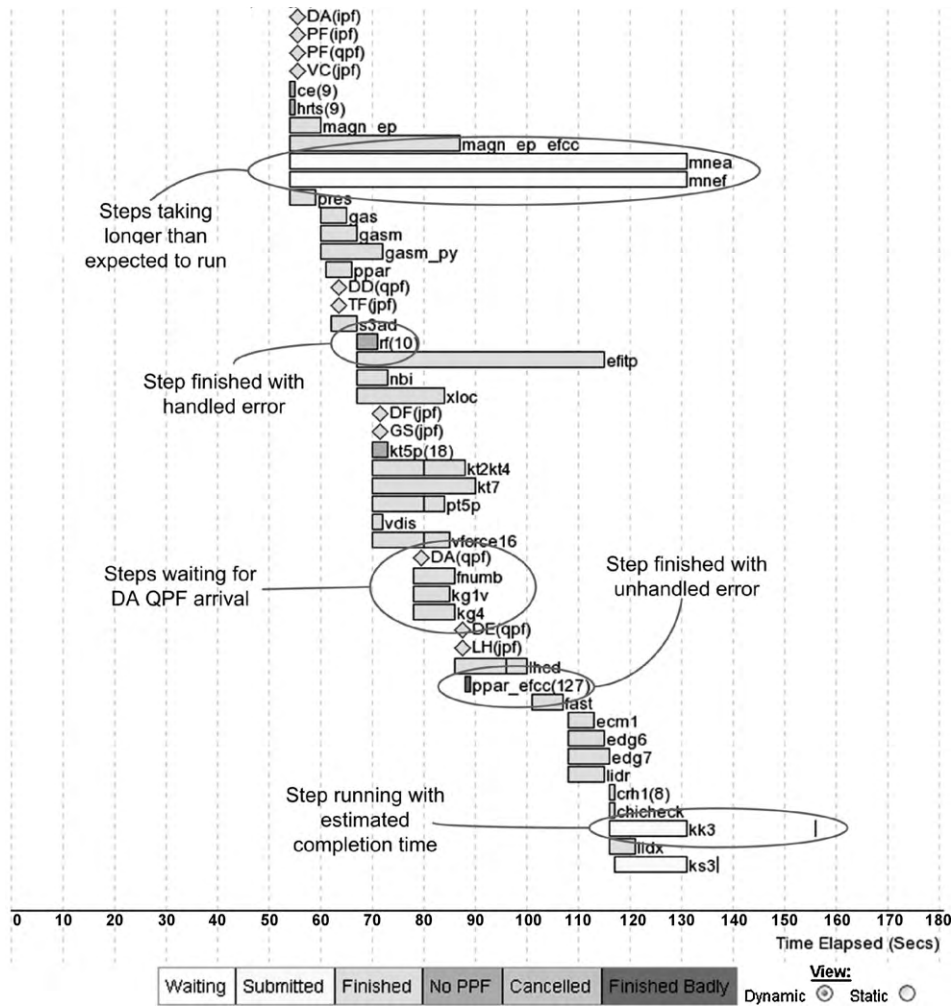


Fig. 6. Annotated output from the new Chain1 monitor.

**4. Phase 2: Chain1 database**

Currently everything to do with the setup, configuration and control of Chain1 is stored in flat files. Updates to input files, source code and dependencies are logged manually. Traceability and audit trails exist but are complex and time consuming to follow. Previous run configurations can be recreated but are resource intensive. Similarly information following a Chain1 run (full or reprocessing job) is stored as flat files in the JET data warehouse [6]. This makes analysis and fault tracing difficult and resource intensive.

Following the release of the new task scheduler described above, work will start on a PostgreSQL [10] database system that will record information about:

- The overall configuration of the current chain. This will be used to create the run environment, dependency rules and configuration of a Chain1 run.
- The history of analysis codes, raw data sources and run configurations, to aid the audit trail and recreate past run environments.
- Every Chain1 run (full or reprocessing) used to create public data. This will (for the first time) include details about the requestor, rationale for the request and performance data which can be used to trace performance over long time periods.

The proposed structure of the database is shown in Fig. 7.

Run data will be cross-referenced against configuration information, so, for example, for any run we can determine code versions and configuration settings used by the steps in it. If an error is discovered in a step (for instance, incorrect calibration), we can determine exactly what data is affected and reprocess only the affected pulses, using the same Chain1 configuration used at the time of the first run, with appropriate corrections.

**5. Phase 3: workflow improvements**

Up to now, we have used our own processing logic to determine which analysis codes can run based on what has happened previously in the chain, and what data is available for use. The master processing node uses rsh commands to submit processing jobs to nodes within the intershot JAC cluster. Once submitted, jobs run independently of the task scheduler.

This is unsatisfactory for two reasons:

- The system is only partially event-driven, as polling for job completion for steps spawned via rsh is processor intensive and inefficient.
- The processing logic engine is highly efficient but difficult to maintain and extend.

We are therefore planning to enhance our workflow. Job submission via rsh will be replaced by a socket-based client-server system.

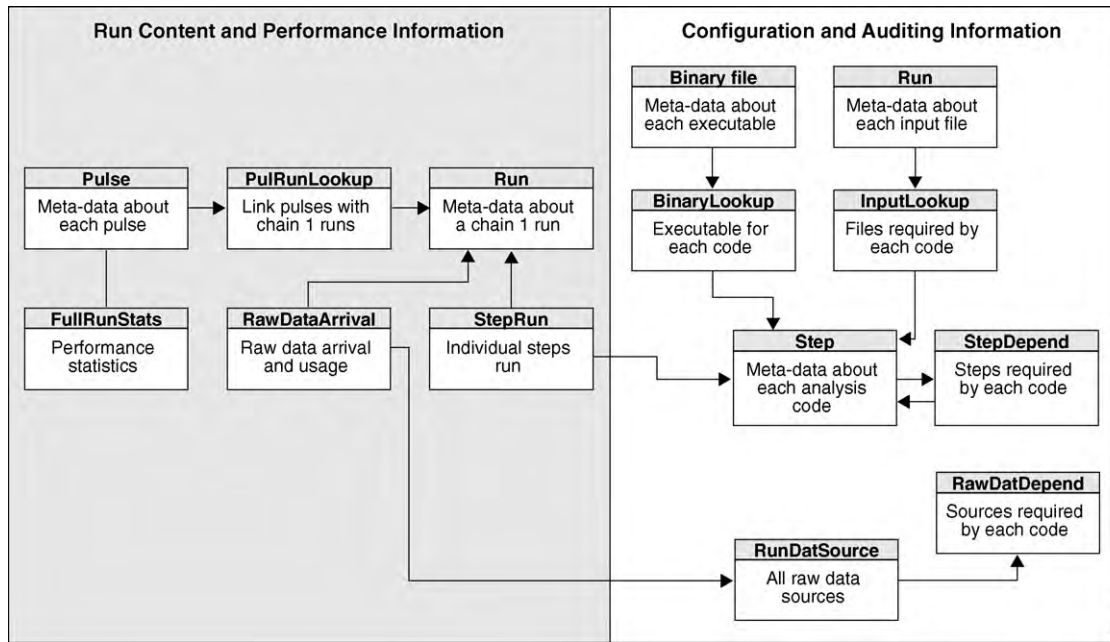


Fig. 7. Proposed structure of Chain1 database.

This will allow two-way communication between the master node and the nodes running steps, so that step completion will become an event which is notified to the master node. The independence of the individual analysis codes will be maintained under this new system, which is illustrated in Fig. 8.

We will consider using a third-party workflow engine to manage data arrivals and job submissions: our favoured approach is a Business Process Execution Language (BPEL) [11] approach using the Apache Axis2/C processing engine [12]. BPEL markup defining the dependency structure for a pulse would be generated based on configuration data stored in the Chain1 database, the resulting file being used by the processing engine to manage the analysis chain for that pulse. However, we need to consider the efficiency of such an approach. If it took a second for the engine to decide what to do next, that would be too slow for the needs of Chain1. If this is the case, we may still use a third party tool to model the processing chain, then develop our existing logic engine to alleviate maintenance issues and ensure future extensibility.

It is important to note that optimisation of the chain will always be a complex problem involving some expert intervention. There is no “typical” JET pulse, and the data arrival times and CPU requirements (particularly for codes which need to iterate to a solution) will differ for different pulse configurations.

This aspect of the work is still at the planning stage, so will not be in use during the Summer 2009 JET experimental campaign.

**6. Phase 4: better support for analysis code developers**

Currently, any changes to an analysis code owned by the physicists and engineers have to come through the Chain1 support team. This includes both code changes and modifications to calibration/configuration data. This can lead to delays in implementing required changes.

Once the planned new control database is in place, we can investigate allowing extended involvement by the analysis code developers. So for example they could:

- automatically upload new calibration and configuration data;
- upload source code and executables for inclusion in the chain;
- retrieve current or old versions of an analysis code for offline data processing;
- view the history of their code – when it changed, what changed in it, when it ran, when it failed.

These concepts would have quality assurance implications, however the database approach will allow us to control which version of an analysis code is used in the production chain, so a new version will not be used before the QA process is completed.

**7. Wider applicability**

The Chain1 processing system has been in place for 10 years and proved extremely reliable. The code management framework within the system has also proved effective, albeit labour intensive. Could the same approach be used elsewhere?

There are several other data processing chains at JET, including modelling and simulation, real-time analysis, and offline detailed analysis. The relationships between these are indicated in Fig. 9. Could the concepts and codebases of these systems be combined so there is only a single product to maintain? Changes to the envisaged Chain1 workflow to support this should be considered in the design of the new system.

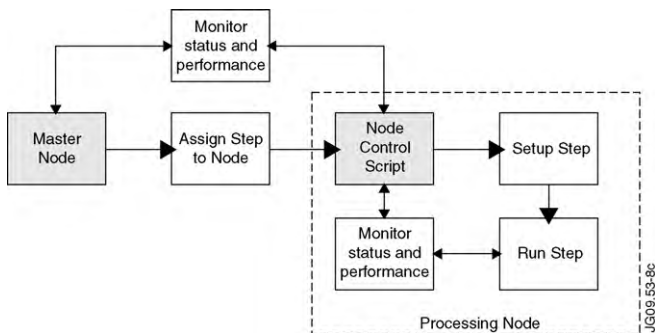


Fig. 8. Proposed client/server system.

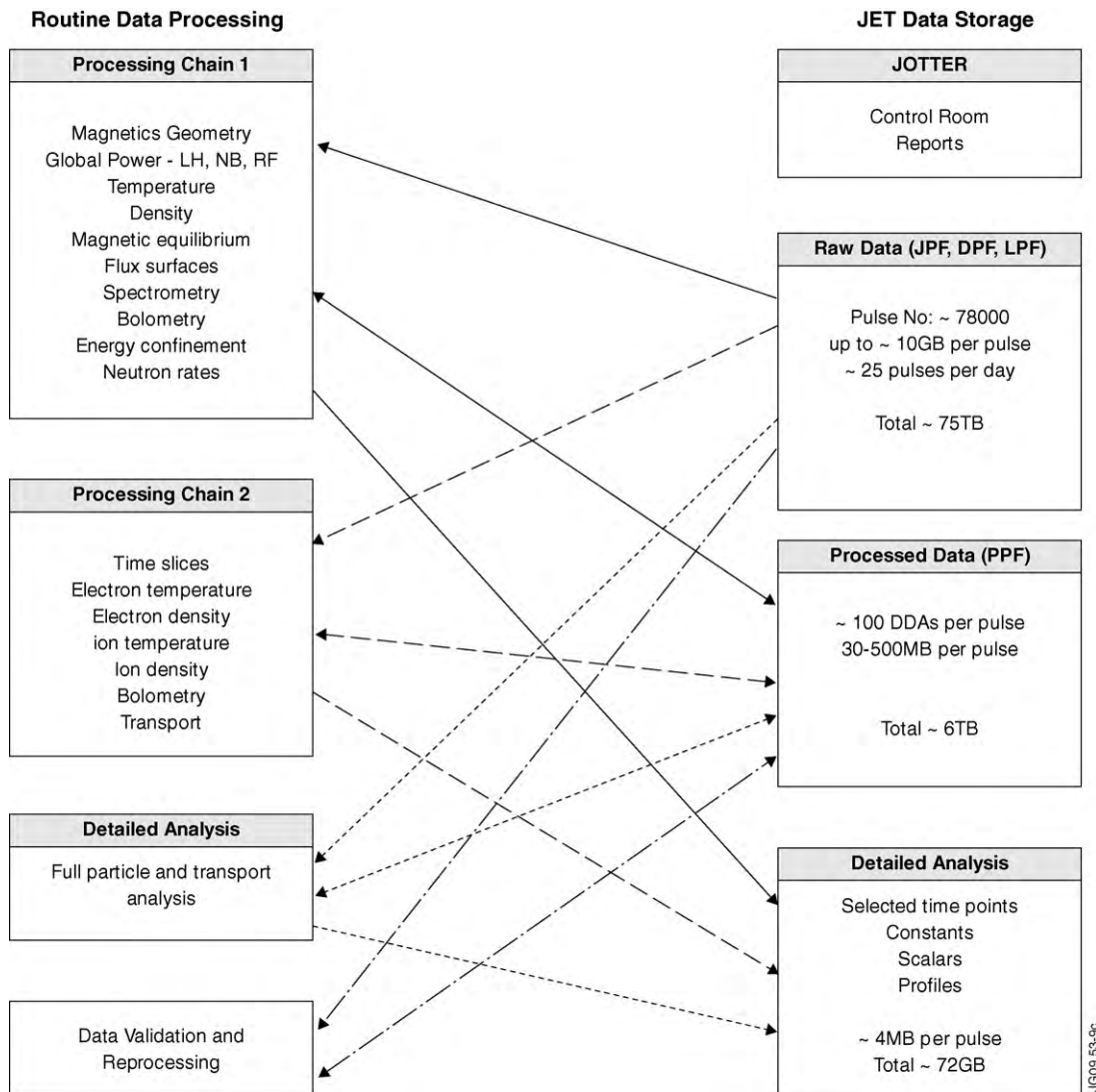


Fig. 9. Chain1 in the context of other JET data analysis system.

In the long-term, it would be beneficial to consider a full data analysis workflow of which Chain1 is just one stage. This would provide consolidated traceability and audit trails for all publicly available JET data, and could, for example, include systems for requesting data validation, links to published articles using the data, and so on.

## 8. Conclusions

JET's Chain1 infrastructure provides a well-proven, stable and efficient system, which has been serving JET operations reliably throughout the project's lifetime. The growing needs of JET have led to new requirements, which are underway. This new infrastructure will provide a useful test case for the data analysis needs of future devices. ITER's equivalent of the Intershot Analysis is likely to be analysis of a segment of a pulse in real time as the pulse executes, but many of the same challenges of scheduling and code integration will apply.

Key lessons learned from JET's experience of Chain1 are:

1. Software quality and integrity of the output data are crucial. Assurance of both should be built in to the development process of new codes. For ITER, there will be stricter QA and regulatory

requirements so a more thorough approach to both these issues needs to be in place, possibly including the use of code coverage tools.

2. An integrated team of computational physicists and software engineers is a recommended approach to maintaining and developing an analysis infrastructure.
3. The full data lifecycle, from modelling and simulation, through real-time analysis, between pulse analysis and later in-depth analysis, should be considered as a single workflow. For ITER, the lines between modelling and simulation, real-time analysis, analysis of a segment of a pulse, and full analysis of a long pulse, are likely to be blurred, so this is a key issue.
4. A flexible framework for integration of external codes is crucial for an international research facility. ITER's requirements will be more complex than JET's due to the larger number of international partners.
5. Data production and analysis complexity are both constantly increasing, so maintaining the efficiency of data analysis requires regular review. The software and hardware infrastructure must both be flexible enough to adapt for future needs. JET's current hardware infrastructure would not have been envisaged 20 years ago, the same may be true of the infrastructure ITER will use in the future.

## Acknowledgements

This work was jointly funded by the United Kingdom Engineering and Physical Sciences Research Council and by the European Communities under the contract of Association between EURATOM and UKAEA. The views and opinions expressed herein do not necessarily reflect those of the European Commission. This work was carried out within the framework of the European Fusion Development Agreement.

## References

- [1] J.W. Farthing, Data Management at JET with a look forward to ITER, <http://accelconf.web.cern.ch/accelconf/ica07/PAPERS/TOPA01.PDF>, ICALEPCS 2007, Knoxville, 2007.
- [2] J.P. Christiansen, Integrated analysis of data from JET, *J. Comput. Phys.* 73 (1987) 85.
- [3] CVS: <http://www.nongnu.org/cvs/>.
- [4] Perl: <http://www.perl.org/>.
- [5] M. Wheatley, M. Rainford, CODAS object monitoring service, *Fusion Eng. Des.* (2001) 56–57.
- [6] R. Layne, M. Wheatley, New data storage and retrieval systems for JET data, *Fusion Eng. Des.* 60 (3) (2002).
- [7] Fedora: <http://fedoraproject.org/>.
- [8] R. Layne, Recent developments in post-pulse information and data storage at JET, in: 5th IAEA TM on Control, Data Acquisition, and Remote Participation, Budapest, 2005.
- [9] PHP: <http://www.php.net/>.
- [10] PostgreSQL: <http://www.postgresql.org>.
- [11] BPEL: <http://www.ibm.com/developerworks/library/specification/ws-bpel/>.
- [12] Apache Axis2/c: <http://ws.apache.org/axis2/c/>.